UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/631,185	07/31/2003	Gerard Chauvel	TI-35431	1444

23494 7590 12/09/2009
TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER	
SAVLA, ARPAN P	
ART UNIT	PAPER NUMBER
2185	
NOTIFICATION DATE	DELIVERY MODE
12/09/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

uspto@ti.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 10/631,185
Filing Date: July 31, 2003
Appellant(s): CHAUVEL ET AL.

Tim D. Chheda
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed September 4, 2009 appealing from the Office action mailed March 16, 2009.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,687,336	Shen et al.	11-1997
7,065,613	Flake et al.	6-2006
6,026,485	O'Connor et al.	2-2000
5,893,121	Ebrahim et al.	4-1999

Kozierok, Charles M., "The Memory Controller", 17 April 2001, The PC Guide, Site Version 2.2.0.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

<u>Claims 11 and 13</u> are rejected under 35 U.S.C. 102(b) as being anticipated by Shen et al. (U.S. Patent 5,687,336) (hereinafter "Shen") with "The PC Guide: The Memory Controller" (hereinafter "PC Guide") offered as extrinsic evidence.

<u>As per claim 11</u>, Shen discloses a computer system, comprising:

a processor (col. 4, line 37; Fig. 2);

a memory coupled to the processor (col. 4, lines 33-34; Fig. 2, element 26); *It should be noted that the memory physically lies within the processor, thus making the memory and processor coupled.*

a stack that exists in memory and contains stack data (col. 4, lines 33-34; Fig. 2, element 26);

a memory controller coupled to the memory (col. 5, lines 19-23). *It should be noted that Shen does not expressly disclose a memory controller in the design, however, PC Guide states that every computer has within it a hardware logic circuit called the memory controller. Also, PC Guide states that the memory controller generates the necessary signals to control the reading and writing of information from and to the memory. Lastly, PC Guide states that the memory controller interfaces the memory with the other major parts of the computer system.*

trend logic (col. 4, lines 40-43; Fig. 2, element 20);

wherein the processor executes instructions (col. 4, lines 36-37);

wherein the trend logic provides trend information about the stack to the controller (col. 5, lines 10-14 and 19-22; Fig. 2, elements 20, 24, 40, and 94); *It should noted that the trend information is calculated by the three-way addition of the stack pointer, segment base, and increment value (which comes from the increment logic) and then the trend information is sent from the three-port adder to the memory controller.*

wherein the trend information about the stack is based on at least one future instruction (col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; Figs. 2 and 4).

<u>As per claim 13</u>, Shen discloses the trend logic determines a net stack trend based on current instruction and future instruction information coming from the decode logic (col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15; col. 4, lines 36-60; Fig. 2, elements 20, 30, and 94).

<u>Claims 17-20</u> are rejected under 35 U.S.C. 102(e) as being anticipated by Flake et al. (U.S. Patent 7,065,613) (hereinafter "Flake").

<u>As per claim 17</u>, Flake discloses a method, comprising:

issuing a write request to a cache memory, wherein the cache memory includes multiple cache lines (col. 5, line 55; col. 6, lines 4-5; Fig. 6);

determining whether the write request refers to a predetermined word within a dirty cache line (col. 5, lines 55-57; Fig. 5). *See the citation note for claim 6 below.*

determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing (col. 6, lines 13-31).

<u>As per claim 18</u>, Flake discloses determining whether the write request will be to the end of a dirty cache line (col. 5, lines 55-57; Fig. 5). *See the citation note for claim 6 above.*

<u>As per claim 19</u>, Flake discloses the stack size is increasing and the dirty cache line is written to a main memory (col. 6, lines 13-28).

<u>As per claim 20</u>, Flake discloses the stack size is decreasing and the dirty cache line is retained in the cache memory (col. 6, lines 29-31).

<u>Claims 1-4, 6-10, 15, and 16</u> are rejected under 35 U.S.C. 103(a) as being unpatentable over Shen in view of Flake with PC Guide offered as extrinsic evidence.

<u>As per claim 1</u>, Shen discloses a method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions (col. 4, lines 11-15 and 36-60; Fig. 2; col. 3, line 65 – col. 4, line 7; col. 6, line 58 – col. 7, line 15; Fig. 4); *It should be noted that Shen's final value for the stack pointer indicates whether the stack size has increased or decreased.*

Shen does not disclose reducing data traffic between various levels of a memory based on the trend information.

Flake discloses reducing data traffic between various levels of a memory based on trend information (col. 6, lines 32-44). *It should be noted that the decision to skip reading the line from main memory is based on the stack operation.*

Shen and Flake are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Flake's miss algorithm to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of reducing the number of accesses to main memory.

As per claim 2, the combination of Shen/Flake discloses determining the trend information includes examining future instructions to determine if the size of the stack is going to decrease as a result of future instructions (Shen, col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; col. 5, lines 1-5 and 9-10; Figs. 2 and 4).

As per claim 3, the combination of Shen/Flake discloses a predetermined number of instructions are used in determining stack trend information (Shen, col. 3, lines 65-67; col. 6, lines 47-48). *It should be noted that if there is only one instruction per stage and there are five stages in the pipeline, then five instructions are used in determining stack trend information.*

As per claim 4, the combination of Shen/Flake discloses the number of predetermined instructions is at least two (Shen, col. 3, lines 65-67; col. 6, lines 47-48). *See citation note for claim 3 above.*

As per claim 6, the combination of Shen/Flake discloses determining which word of the dirty cache line is going to be written to (Flake, col. 5, lines 55-57; Fig. 5). *It should be noted that in Flake's cache each line contains 2 bytes (col. 5, line 48). Since 2 bytes = 1 word, it follows that in Flake's cache each line contains 1 word. Thus, once the algorithm chooses which dirty line to replace, as a consequence it has also determined which word of the dirty line (to be replaced) is going to written to.*

As per claim 8, the combination of Shen/Flake discloses determining the trend information includes examining future instructions to determine if the size of the stack is going to increase as a result of future instructions (Shen, col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; col. 5, lines 1-7; Figs. 2 and 4).

As per claim 7, Shen discloses a method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions (col. 4, lines 11-15 and 36-60; Fig. 2; col. 3, line 65 – col. 4, line 7; col. 6, line 58 – col. 7, line 15; Fig. 4); *See the citation note for the same limitation in claim 1 above.*

utilizing the trend information to reduce data traffic between various levels of a memory (col. 8, lines 34-47). *It should be noted that the limitation "to reduce data traffic between various levels of a memory" is merely a recitation of intended use for the "trend information." A recitation of the intended use of the claimed invention must result in a structural difference between the claimed invention and the prior art in order to* *patentably distinguish the claimed invention from the prior art. If the prior art structure is capable of performing the intended use, then it meets the claim. It should also be noted that by having the stack pointer signal a mis-aligned access, push/pop pairing is prevented, a second cache access is prevented, thus, data traffic between various levels of the cache are reduced.*

Shen does not disclose the levels of memory comprise a cache memory containing multiple cache lines and a main memory, and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing.

Flake discloses the levels of memory comprise a cache memory containing multiple cache lines and a main memory (col. 6, lines 4-9; Fig. 6), and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing (col. 6, lines 29-31). *It should be noted that a "deletion of a group of stack objects" is analogous to the "stack decreasing."*

Shen and Flake are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Flake's miss algorithm to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of reducing the number of accesses to main memory.

<u>As per claim 9</u>, Shen discloses a method of managing memory, comprising:

determining stack trend information using current and future stack operating instructions (col. 4, lines 11-15 and 36-60; Fig. 2; col. 3, line 65 – col. 4, line 7; col. 6, line 58 – col. 7, line 15; Fig. 4), including determining if the size of the stack is going to increase as a result of future instructions (col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; col. 5, lines 1-7; Figs. 2 and 4);

utilizing the trend information to reduce data traffic between various levels of a memory (col. 8, lines 34-47). *See the citation note for the same limitation in claim 7 above.*

Shen does not disclose determining if a line is written back including analyzing the trend information and including examining a dirty cache line to determine which word of the dirty cache line is going to be written to.

Flake discloses determining if a line is written back including analyzing the trend information (col. 6, lines 13-28) and including examining a dirty cache line to determine which word of the dirty cache line is going to be written to (col. 5, lines 55-57; Fig. 5). *See the citation note for claim 6 above.*

Shen and Flake are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Flake's miss algorithm to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of reducing the number of accesses to main memory.

As per claim 10, the combination of Shen/Flake discloses the dirty cache line is written from a cache memory to a main memory (Flake, col. 6, lines 22-23).

As per claim 15, Shen discloses a computer system, comprising:

a processor (col. 4, line 37; Fig. 2);

a memory coupled to the processor (col. 4, lines 33-34; Fig. 2, element 26); *See the citation note for the same limitation in claim 11 above.*

a stack that exists in memory and contains stack data (col. 4, lines 33-34; Fig. 2, element 26);

a memory controller coupled to the memory (col. 5, lines 19-23). *It should be noted that Shen does not expressly disclose a memory controller in the design, however, PC Guide states that every computer has within it a hardware logic circuit called the memory controller. Also, PC Guide states that the memory controller generates the necessary signals to control the reading and writing of information from* *and to the memory. Lastly, PC Guide states that the memory controller interfaces the memory with the other major parts of the computer system.*

trend logic (col. 4, lines 40-43; Fig. 2, element 20);

wherein the processor executes instructions (col. 4, lines 36-37);

wherein the trend logic provides trend information about the stack to the controller (col. 5, lines 10-14 and 19-22; Fig. 2, elements 20, 24, 40, and 94); *See the citation note for the same limitation in claim 11 above.*

wherein the trend information about the stack is based on at least one future instruction (col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; Figs. 2 and 4).

Shen does not disclose the memory includes a cache memory containing multiple cache lines and a main memory, and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing.

Flake discloses the memory includes a cache memory containing multiple cache lines and a main memory (col. 6, lines 4-9; Fig. 6), and wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing (col. 6, lines 29-31). *See the citation note for the similar limitation in claim 7 above.*

Shen and Flake are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Flake's miss algorithm to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of reducing the number of accesses to main memory.

As per claim 16, Shen discloses a computer system, comprising:

a processor (col. 4, line 37; Fig. 2);

a memory coupled to the processor (col. 4, lines 33-34; Fig. 2, element 26); *See the citation note for the same limitation in claim 11 above.*

a stack that exists in memory and contains stack data (col. 4, lines 33-34; Fig. 2, element 26);

a memory controller coupled to the memory (col. 5, lines 19-23). *It should be noted that Shen does not expressly disclose a memory controller in the design, however, PC Guide states that every computer has within it a hardware logic circuit called the memory controller. Also, PC Guide states that the memory controller generates the necessary signals to control the reading and writing of information from and to the memory. Lastly, PC Guide states that the memory controller interfaces the memory with the other major parts of the computer system.*

trend logic (col. 4, lines 40-43; Fig. 2, element 20);

wherein the processor executes instructions (col. 4, lines 36-37);

wherein the trend logic provides trend information about the stack to the controller (col. 5, lines 10-14 and 19-22; Fig. 2, elements 20, 24, 40, and 94); *See the citation note for the same limitation in claim 11 above.*

wherein the trend information about the stack is based on at least one future instruction (col. 3, line 65 – col. 4, line 7; col. 4, lines 11-15 and 36-60; Figs. 2 and 4).

Shen does not disclose the memory includes a cache memory and a main memory, and wherein the cache memory contains a dirty cache line, and wherein the dirty cache line is written to main memory if the trend information indicates the stack is increasing.

Flake discloses the memory includes a cache memory and a main memory (col. 6, lines 4-9; Fig. 6), and wherein the cache memory contains a dirty cache line (col. 6, lines 19-20), and wherein the dirty cache line is written to main memory if the trend information indicates the stack is increasing (col. 6, lines 13-28). *It should be noted that "stack growth" is analogous to the "stack increasing."*

Shen and Flake are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Flake's miss algorithm to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of reducing the number of accesses to main memory.

<u>Claims 12 and 14</u> are rejected under 35 U.S.C. 103(a) as being unpatentable over Shen in view of O'Connor et al. (U.S. Patent 6,026,485) (hereinafter "O'Connor").

<u>As per claim 12</u>, Shen discloses all the limitations of claim 12 except an instruction decoder comprising a first portion that decodes current instructions and a second portion that decodes future instructions.

O'Connor discloses an instruction decoder comprising a first portion that decodes current instructions and a second portion that decodes future instructions (col. 3, lines 5-10, 15-18, and 42-48). *It should be noted that first instructions are current instructions while second instructions are future instructions.*

Shen and O'Connor are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply O'Connor's instruction decoder to Shen's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of instruction folding for a stack based machine.

<u>As per claim 14</u>, the combination of Shen/O'Connor discloses the second portion of the decoder is adjusted so that the number of future instructions that are decoded equals at least two (O'Connor, col. 3, lines 59-60). *It should be noted that first instructions are current instructions while second and third instructions are both future instructions.*

<u>Claim 5</u> is rejected under 35 U.S.C. 103(a) as being unpatentable over Shen in view of Flake as applied to claim 1, and further in view of Ebrahim et al. (U.S. Patent 5,893,121) (hereinafter "Ebrahim").

<u>As per claim 5</u>, the combination of Shen/Flake discloses all the limitations of claim 5 except the cache memory maintains a single dirty cache line for stack data.

Ebrahim discloses the cache memory maintains a single dirty cache line for stack data (col. 5, lines 32-35). *It should be noted that a cache block maintaining a dirty bit is analogous to a "dirty cache line".*

The combination of Shen/Flake and Ebrahim are analogous art because they are from the same field of endeavor, that being computer memory systems.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to apply Ebrahim's stack cache memory containing a single dirty cache line to Shen/Flake's stack trend tracker system because all the claimed elements were known in the prior art and one skilled in the art could have combined the elements as claimed by known methods with no change in their respective functions, and the combination would have yielded the predictable results of automatically recovering memory objects no longer in use by the operating system and application programs in a computer system.

(10) Response to Argument

Response to B.1.

Appellant argues, in section B.1., that:

"Independent claim 1 recites, in part, "reducing data traffic between various levels of a memory based on the [stack] trend information." Independent claims 7 and 9 recite a similar limitation: "utilizing the [stack] trend information to reduce data traffic between various levels of a memory." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation."

The Examiner respectfully disagrees. As detailed in the previous Office actions, as commonly defined, a "trend" is "the general movement over time of a statistically detectable change". In Shen, a new stack pointer indicates the final value of the stack and therefore the new stack pointer reflects the general movement of the stack size over a period of time, the period of time being the time between the generation of the old stack pointer (SP) and the generation of the new stack pointer (SP'). The various stages in Shen's pipelined processor contain both current and future instructions. These instructions are used to generate a new stack pointer. As detailed above, the new stack pointer reflects the “trend" of the stack and is therefore equivalent to "trend information", as simply and broadly claimed by Appellant.

In Flake, col. 6, lines 32-44 describe how reduction of data traffic between various levels of a memory is based on various stack operations. When there is a deletion of a group of stack objects (i.e. the stack is decreasing), or in other words a stack pop operation, no cache functions occur (see col. 6, lines 29-31 of Flake). Such a course of action makes sense because when data is deleted (i.e. popped) from a stack the data is effectively invalidated. Since the data is invalid, it can simply be discarded. There is no need to write invalid data back to main memory, thus reducing traffic between various levels of memory. When there is a stack-growth write (i.e. the stack is increasing), or in other words a stack push operation, reading a line from main memory is skipped. Such a course of action also makes sense because the operation corresponds to the unused section of the stack, thus, the data being written (i.e. pushed) to the stack is new. Since the data being written is new, it does not correspond to any data currently in main memory, therefore, there is no need to read any data from main memory in order to complete the stack-growth write. As a result, reading a line from main memory is skipped, thus reducing traffic between various levels of memory. The Examiner also notes that Flake discloses a broad form of “stack trend information” as well because a deletion of a group of stack objects shows a decreasing trend in stack size while a stack-growth write shows an increasing trend in stack size.

It appears Appellant is attacking the references individually, however, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). The rejection of claims 1, 7, and 9 is not based on individual references, but rather the combination of Shen and Flake. When combining Shen and Flake, in the manner set forth by the Examiner in the rejection above, Flake's miss algorithm is applied to Shen's stack trend tracker system. Such a combination would allow a reduction of data traffic between various levels of a memory (such as no cache operations being performed or skipping reads from main memory) based on the value of the new stack pointer (i.e. the trend information).

Additionally, with respect to claims 7 and 9, the Examiner notes that the term "to" renders the "reduce data traffic between various levels of a memory" limitation as merely a recitation of intended use of the claimed trend information. A recitation of the intended use of the claimed invention must result in a structural difference between the claimed invention and the prior art in order to patentably distinguish the claimed invention from the prior art. If the prior art structure is capable of performing the intended use, then it meets the claim. See MPEP 2106 II(C) and 2111.04. In Shen, when the stack pointer signals a mis-aligned access, push/pop pairing is prevented, and therefore a second cache access is prevented. Preventing a cache access will in turn reduce data traffic between various levels of memory. Accordingly, based on the foregoing, the combination of Shen/Flake with PC Guide offered as extrinsic evidence renders claims 1-4 and 6-10 unpatentable.

Response to B.2.

Appellant argues, in section B.2., that:

"Independent claim 15 recites, in part, "wherein the trend information is used to restrict writing dirty cache lines from cache memory to main memory when the trend information indicates the stack is decreasing." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation."

The Examiner respectfully disagrees. Col. 6, lines 29-31 of Flake state:

> "The next processor operation is a <u>deletion of a group of stack objects</u> achieved by changing the stack pointer to A8. <u>No cache functions occur</u>." (emphasis added)

As is clear from the cited text above, Flake's deletion of a group of stack objects discloses the stack is decreasing. When there is a deletion of a group of stack objects (i.e. the stack is decreasing), no cache functions occur (see col. 6, lines 29-31 of Flake). When no cache functions occur, there are no reads or writes between cache and main memory. Thus, no cache functions occurring inherently anticipate restricting the writing dirty cache lines from cache memory to main memory, because no cache functions occurring means there are no transactions between cache and main memory in general. Another way to look at it is when data is deleted (i.e. popped) from a stack the data is effectively invalidated. Since the data is invalid, it can simply be discarded. There is no need to write invalid data back to main memory. Accordingly, based on the foregoing, the combination of Shen/Flake with PC Guide offered as extrinsic evidence renders claim 15 unpatentable.

Response to B.3.

Appellant argues, in section B.3., that:

“Independent claim 16 recites, in part, "wherein the dirty cache line is written to main memory if the trend information indicates the stack is increasing." However, Examiner erred in rejecting the claim because the cited references fail to teach or suggest the limitation.”

The Examiner respectfully disagrees. Col. 6, lines 13-23 of Flake state:

> The first processor operation is a stack-growth write to address 9F (for a downward growing stack shown in FIG. 6), which begins by writing to that address and then setting the stack pointer to 9F. The cache logic determines that this write is a miss by virtue of its tag value, binary 10011, and index value, binary 1, not matching line 1. The miss begins the line replacement algorithm of FIG. 5. This line is dirty and the old tag value of binary 00000, corresponding to addresses 04 07, does not point to the unused section of the stack. This forces a write of the non-stack data of addresses 04 07 from the cache to the main memory. (emphasis added)

As is clear from the cited text above, Flake's stack-growth write discloses the stack is increasing. As is also clear from the cited text, the write causes a miss. That miss in turn begins the line replacement algorithm in which a dirty line is written from cache to main memory. Thus, it follows that the stack-growth write (i.e. the stack increasing) causes the dirty line to be written from cache to main memory. Therefore, contrary to Appellant's allegation, there is indeed a causal relationship between the stack-growth write and writing a dirty cache line back to main memory. Accordingly, based on the foregoing, the combination of Shen/Flake with PC Guide offered as extrinsic evidence renders claim 16 unpatentable.

Response to C.

Appellant argues, in section C., that:

"…O'Connor fails to teach the deficiencies of Shen. Therefore, the reasoning below applies, and claims 12 and 14 are allowable over Shen in view of O'Connor."

The Examiner respectfully disagrees and refers Appellant below to the response to E. which details how Shen discloses the trend information about the stack is based on at least one future instruction. Accordingly, the combination of Shen/O'Connor with PC Guide offered as extrinsic evidence renders claims 12 and 14 unpatentable.

Response to D.

Appellant argues, in section D., that:

"…Ebrahim fails to teach the deficiencies of Shen. Therefore, the reasoning above applies, and claim 5 is allowable over Shen in view of Flake in further in view of Ebrahim."

The Examiner respectfully disagrees and refers Appellant above to the response to B.1. which details how the combination of Shen/Flake renders claim 1 unpatentable. Accordingly, the combination of Shen/Flake/Ebrahim with PC Guide offered as extrinsic evidence renders claim 5 unpatentable.

Response to E.

Appellant argues, in section E., that:

“Independent claim 11 recites, in part, "wherein the trend information about the stack is based on at least one future instruction." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation.”

The Examiner respectfully disagrees. Col. 4, lines 40-44 of Shen state:

> “Pipeline valid bits 50 indicate the <u>locations of valid stack instructions in the pipeline. From the locations of the stack instructions</u> from valid bits 50, increment logic 20 determines increment value 94 to add to stack pointer 12 stored in register file 10.” (emphasis added)

The Examiner also refers Appellant to Fig. 3 as well as col. 5, lines 1-18 of Shen which state:

> The increment value 94 is determined by increment logic 20 by reading stack valid bits in pipeline valid bits 50. <u>Stack valid bits are read by increment logic 20 for the D, A, C, M, and W stages. The stack valid bits indicate the locations of valid stack instructions</u>. As shown in FIG. 3, push instructions, which increment the stack pointer by -4, are located in the D and C stages. In the C stage two push instructions are present, so the stack pointer must be incremented by a double amount, -8. A pop instruction is present in stage M, which decrements the stack pointer by +4. The net result: -4+-8++4=-8 is increment value 94 outputted from increment logic 20. It is the overall displacement to add to the stack pointer. This addition may be implemented in increment logic 20 as a small 3-bit 5-port adder, or preferably as combinatorial logic.

As can be seen from the cited portions of Shen above along with Fig. 3, stack instructions from the C, M, and W stages are used to determine increment value 94, which is input into adder 40 in stage A. The Examiner submits that relative to stage A, stages C, M, and W are all "future" stages in the pipeline, and therefore contain "future instructions" relative to stage A. The three future instructions from stages C, M, and W are all used to determine the output of adder 40. The value of the new stack pointer (SP’) (i.e. the trend information) that is calculated in stage M is based on the output of adder 40. Therefore, it follows that the value of the new stack pointer (SP') (i.e. the trend information) is based on the three future instructions from stages C, M, and W. As a result, Shen sufficiently discloses the trend information about the stack is based on at least one future instruction. Accordingly, based on the foregoing, Shen with PC Guide offered as extrinsic evidence renders claims 11 and 13 unpatentable.

Response to F.

Appellant argues, in section F., that:

"Independent claim 1-7 recites, in part, "determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing." However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitation."

The Examiner respectfully disagrees and refers Appellant above to the responses to B.2. and B.3. As detailed in the response to B.2., Flake's deletion of a group of stack objects discloses the stack is decreasing. When there is a deletion of a group of stack objects (i.e. the stack is decreasing), no cache functions occur (see col. 6, lines 29-31 of Flake). When no cache functions occur, there are no reads or writes between cache and main memory. Thus, no cache functions occurring inherently anticipate restricting writing dirty cache lines from cache memory to main memory, because no cache functions occurring means there are no transactions between cache and main memory in general. Another way to look at it is when data is deleted (i.e.

popped) from a stack the data is effectively invalidated. Since the data is invalid, it can simply be discarded. There is no need to write invalid data back to main memory.

As detailed in the response to B.3., Flake's stack-growth write discloses the stack is increasing. It should also be noted that the write causes a miss. That miss in turn begins the line replacement algorithm in which a dirty line is written from cache to main memory. Thus, it follows that the stack-growth write (i.e. the stack increasing) causes the dirty line to be written from cache to main memory. Therefore, contrary to Appellant's allegation, there is indeed a causal relationship between the stack-growth write and writing a dirty cache line back to main memory. As a result, Flake sufficiently discloses determining whether to write the dirty cache line to main memory based on whether the size of a stack is increasing or decreasing. Accordingly, based on the foregoing, Flake renders claims 17-20 unpatentable.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Arpan Savla/

Examiner, Art Unit 2185

Conferees:

/Kevin L Ellis/
Supervisory Patent Examiner, Art Unit 2117

/Sanjiv Shah/
Supervisory Patent Examiner, Art Unit 2185